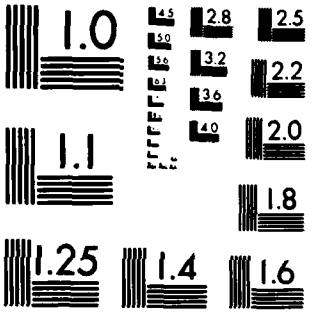
AD-A127 169 CONVERSION OF ALGORITHMS TO CUSTOM INTEGRATED CIRCUIT 1/1
DEVICES(U) MASSACHUSETTS INST OF TECH CAMBRIDGE
RESEARCH LAB OF ELECTRONICS J ALLEN 30 DEC 82
UNCLASSIFIED AFOSR-TR-83-0248 F49620-81-C-0054 F/G 9/3 . NL

END
DATE FILMED
5 83
DTIC



AFOSR-TR- 83 0218

AD A127169

FINAL REPORT

Conversion of Algorithms to Custom Integrated Circuit Devices

AFOSR Contract F49620-81-C-0054

covering the period

15 March 1981 - 14 May 1982

submitted by

Jonathan Allen

Massachusetts Institute of Technology
Research Laboratory of Electronics
Cambridge, MA 02139

83 04 20 209

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 82-0248	2. GOVT ACCESSION NO. AD A127169	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) CONVERSION OF ALGORITHMS TO CUSTOM INTEGRATED CIRCUIT DEVICES		5. TYPE OF REPORT & PERIOD COVERED FINAL REPORT 3/15/81 - 5/14/82
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Jonathan Allen		8. CONTRACT OR GRANT NUMBER(s) F49620-81-C-0054
9. PERFORMING ORGANIZATION NAME AND ADDRESS Research Laboratory of Electronics Massachusetts Institute of Technology Cambridge, MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 2305/B3
11. CONTROLLING OFFICE NAME AND ADDRESS AFOSR Bolling AFB, Washington, DC 20332		12. REPORT DATE December 30, 1982
		13. NUMBER OF PAGES 30
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution limited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

VLSI design, Custom Integrated Circuit Design

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

In this report we gives an overview, work statement, and status of research for the contract Conversion of Algorithms to Custom Integrated Circuits. The basic conceptual framework for this work is to recognize that the design of custom integrated circuits involves the specification of a number of different levels of representation that are qualitatively as well as quantitatively different, and that there is the necessity to derive a set of transformations between these representations so that the complete design can be specified.

(continued)

DD FORM 1 JAN 73 1473

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Accordingly, the work described in this report deals with ways in which these representations can be established as well as the means of deriving one representation from others.

An overview of the entire project is given, followed by detailed description of several different projects focused on the different levels of representation. These include artwork analysis, circuit characterization, logic checking, the design for several application domains, and comments relative to the changing technological base. In the artwork area design rule checking is performed in terms of two different underlying representations of the mask layout specification, and to special architectures derived involving four custom integrated circuits for high performance design rule checking. In the area of interactive graphic layout, three different graphic layout editors are described along with the various features associated with them. In the placement and routing area, channel routers are discussed in the context of the overall PI system for placement and interconnect of arbritrary rectangular cells with interconnect on all four sides.

In the circuit area a variety of concerns are addressed but two dominant projects involve the bounding of delay through interconnect and the high performance extraction of circuit representation from the detailed mask specification. Both of the projects are aimed at the characterization of circuit performance for high performance designs.

In the logic area the MOSSIM unit delay switch level logic simulator is described, together with the underlying theory of MOS digital systems that has been motivated by the needs for this simulation.

Finally, complex applications, both in the signal processing area and in the area of compiled microprocessor base designs are discussed. Experience with these designs is used to motivate a discussion of the way in which all of the above programmatic capability can be interrelated into a cohesive design system.

TABLE OF CONTENTS

I. Overview 1

II. Statement of Work 7

III. Status of Research 15

IV. Publications 25

V. Professional Personnel 26

OVERVIEW

In order to place the research done under this contract in perspective we start by describing the view of integrated circuit design that motivates our work. At the top level the goal is to be able to design custom integrated circuits correctly, quickly, and inexpensively. Custom integrated circuit design has a lot of competition, including readily available commercial parts, and increasingly prevalent semicustom techniques, such as standard cell approaches and the use of gate arrays. If custom design is to continue as a viable approach it is important to be able to meet this competition in an effective way. If a good set of design tools is available for such design, then it need not be any more expensive to produce a design for a custom circuit than for those designs that utilize standard cells or gate arrays. This last statement may seem contradictory, since obviously a great deal of design effort has gone into the creation of standard cells and into the provision of gate array designs. What is really at issue is the question of what fundamental representations the designer manipulates, and when decisions have to be made in order to fix the final design. A major tenet of our approach is that it is possible to capture in programmatic form a great deal of design information that can be used flexibly in many different environments, rather than being restricted to some rigid architectural scheme. Furthermore, by chosing these representations carefully, it is possible to delay the binding of many decisions in a way that helps optimize the layout of a particular design. We want to retain the advantages of canonical forms, such as those used in standard cells and gate arrays, but we want to gain flexibility in the shaping of these cells to provide high performance, and in the way in which they are interconnected in clusters for a particular design application. Just as standard cell design seeks to build on the accumulated experience of designing for many different applications, our custom design approach also seeks to benefit from many different concrete designs. Rather than capturing the designer's expertise in terms of fixed final designs, however, we are trying to capture the designer's expertise in more abstract terms, by discovering and utilizing fundamental frameworks that can represent the designer's intent coupled with the means to partially specify those decisions that have been bound at any given point of the design process. Computer-aided design systems must aid the designer in this way, and provide him with a means to explore a variety of different design options, and to

probe the consequences of these decisions. We also believe that many approaches to integrated circuit design utilize a "take what you get" approach to performance considerations. That is, a design is made and then it is inspected for design parameters such as area, speed, and power dissipation. We want to place these considerations more centrally in design, and accordingly we are exploring the ways in which performance variables can be manipulated in design, and incorporated into programmatic procedures for final mask layout specification. A good example of the opportunities available to the custom integrated circuit designer is afforded by a comparison between gate array designs and custom designs for a particular application. Some industrial practitioners estimate that a gate array design can be as much as four times bigger than a corresponding custom design for the same application. The reason for this large expansion in area is that gate arrays force decisions to be made relatively early in the design process in a way that leads to substantial inefficiencies in the design. For example, it is not uncommon for the gate utilization on a gate array to be no more than sixty-five percent, leaving a third of the cells unused. Also, individual transistors have to be oversized in order to make sure that performance speed criteria are met. Also, routing channels are generally made bigger than they need to be since the gate array must accommodate many different designs. In todays' technology, gate arrays are certainly attractive for many applications because of the low risk involved and the rapid turn-around provided. We believe, however, that for many application areas, such as digital signal processing, we will be able to provide design techniques that will compete effectively with these semicustom approaches. This is a large challenge, but as we understand further the nature of design and the balance between preserving the designer's degrees of freedom versus the use of standard predesigned (yet parameterizable) forms, that we will be able to offer design techniques for high performance systems that will meet all of our basic criteria. Exploring this balance between the designer's efforts and those efforts that can be performed by a good computer-aided design system is a major focus of our work.

In many ways the design of highly complex custom integrated circuits is similar to the design of large programs. We look for identifiable modules that can be connected into a appropriate hierarchy that provides internal structure to the overall design, whether it be in hardware or software. Even

though a lot of our emphasis in this contract is on programmatic techniques, it is important to realize a fundamental difference between the design of large software systems and the design of large custom integrated circuits. In the case of software, the representations used at all levels of the hierarchy and within all modules are identical, namely those provided by the programming language being used. Thus, techniques that seek to enforce well-formedness and correctness are uniform over all levels of the hierarchy within the software system. The situation is very different, however, for hardware design. In the case of the design of custom integrated circuits we actually have two different kinds of hierarchy to deal with. The first kind of hierarchy is, in fact, the same as that already mentioned for software. But there is an additional aspect of hierarchy which is in some sense "orthogonal" to this previously mentioned hierarchy that leads to a number of different levels of representation for the same basic design. These levels of representation can be characterized as the top level functional characterization of the intended behavior, the architecture used to provide the basic computational framework, the logic specification within this architecture, the circuits utilized to realize this logic, the devices that form the active and passive elements within these circuits, and finally the geometrical layout that specifies the detailed mask information needed for integrated circuit fabrication. What is striking about all of these representations is that they are so very different in form. The way in which transistors connect together at the circuit level gives rise to entirely separate considerations than the way in which logic gates are connected together to perform logic functions. Nevertheless, if we perturb one of these representations, it has consequences at all of the other levels of representation. Furthermore, it is important that each of these levels of representation be important to the design process in that each representation deals with a set of issues that are important to the overall design. For example, the detailed specification of logic is clearly important to the computational specification of any design, and so is the circuit performance. When a designer visits each of these representations, then, he is focusing on one particular point of view of the overall design. He knows that there are many other aspects to the design, but it is difficult for him to deal with all of them simultaneously. A good design system must thus provide not only appropriate and insightful representations that must be specified and

manipulated during the design process, but it must also provide for transformations between these representations so that they stay "aligned" during the design process. All of this must be accomplished in the context of very complex designs involving half a million or more transistors.

From the above discussion, we can appreciate that design involves two kinds of tasks. First, the design must be specified, but then it must be verified at each of the several levels of representation that we have mentioned. These two tasks suggest that the overall design procedure can be characterized in terms of an ongoing tension between synthesis and analysis. That is to say, much of the current design of custom integrated circuits consists of the specification of a design, usually instantiated without explicit use of design constraints, followed by an analysis procedure that is intended to verify that the requisite constraints at each representational level are satisfied. We would like to, of course, achieve correctness from the start, without having to verify it after the specification phase. There has been a certain amount of success in this area, such as in the use of PLA generators. These are programs that generate the detailed layout for a programmed logic array which is "correct by construction". We want to learn from these experiences so that performance constraints as well as fundamental design constraints can be incorporated into the basic synthetic or specification procedures. It will take a considerable amount of experience with both specification and verification before we can see how to make this combination take place in an optimal way. Accordingly, a large portion of our effort is devoted to providing the tools for exploring a variety of different design synthesis strategies, coupled with a powerful set of performance evaluation tools that provide the designer with feedback as to the efficacy of a synthetic procedure. This means that we combine intimately within this project both the creation of design tools and the utilization of these tools to design practical chips. Two different categories of chips have served as major foci for our design effort. In one case, we have designed a complex signal processing chip that raises many issues for the creation of digital signal processing implementations in general. These considerations include the provision of substantial parallelism, the implementation of fast arithmetic capability (including multipliers), novel memory circuit design including multiport access, and real time input/output capability for connection to real world devices. An important area of custom integrated circuit design is the

provision of a high level functional specification, and we have worked for some time on the specification of languages for signal processing that can serve as input to the overall design of custom integrated circuits for signal processing. The utilization of such languages is important for capturing the designer's intent in a precise way, since they make explicit the semantic attributes of the design, and provide the designer with clean constructs upon which to base his specification. Another exciting area related to the design of circuits for digital signal processing is the growth of the size of the basic computing elements. These elements have grown from the basic arithmetic-logic capability, through the provision multiplier-accumulator units and now on to matrix-orientated techniques that provide great parallel power and generality and which have given rise to new and novel architectures such as systolic arrays. Modern signal processing gives rise to important new insights into the optimal form of filters, and clearly there is a large interaction between these investigations and the implementation technology which we use for design. We are now bringing into the activities of this contract increased consideration of these factors, and believe that it will enable us to construct highly optimized design procedures for modern signal processing structures. Thus the effective combination of talents from signal processing theory, control theory, computational architecture, and circuit design, are all needed to provide contemporary signal processing systems.

The other major class of systems that we are focusing our attention on are those that are specifiable by some form of basic computational engine that is controlled through the execution of microcode statements. This class of systems is not mutually exclusive with the signal processing designs previously described, but is more appropriate to the basic concerns of language interpreters. Here at MIT we have a long history of emphasis and expertise in the design of various dialects of the programming language LISP and their interpreters. Fundamental knowledge of the nature of these languages, together with the requirements for effectively interpreting them, can be utilized to design optimal computational engines for their execution. The SCHEME dialect of LISP has received a great deal of attention here, and its designers have produced a second generation chip that depends on a great deal of software for compiling the microcode specification on down to the detailed chip layout. The requirements for these designs are specified in terms of arithmetic capability together with specialized register files

interconnected by local bus schemes that provide for a great deal of parallelism. Little emphasis so far has been placed on circuit performance, but rather on architectural strategies for efficiently implementing the language processor. A great deal of useful experience has been obtained from this work, both as regards procedural representations and the compilation strategies for achieving the final design. From these comments it can be realized that the two major foci of chip design contemplated within this contract have been used as motivators for the specification of design tools. We see this as an important attribute of the contract since the provision of design tools without a corresponding user community is a vacuous exercise.

The relationship of underlying technology to the design process is another consideration that is receiving increased attention as part of our studies. This emphasis has arisen in two interesting ways. In one case, we have been concerned with the conversion of existing designs in one technology into an alternative technology. An instance of such a case is the change of a design from a TTL base to a custom MOS circuit. We have come to realize that each of these underlying technologies carries with it a set of useful circuit forms that do not map easily one onto the other. Thus, an important point of view that has been derived from our investigations is that architectures in one technology cannot be simply mapped onto another technology, but instead the basic computational task needs to be reexamined in the context of the implementation technology each time. Even though the experience with these techniques has thus far been limited to a few specific cases, a great deal of insight into the overall design process in each of these technologies has been realized. Another aspect of technological change has been the gradual movement from NMOS into CMOS. We have made sure that all of our design tools operate properly for both NMOS and CMOS, and once again it has been very instructive to discover how different designs can be realized in both NMOS and CMOS and the costs associated with each design. The circuit techniques used in each case vary a great deal, and give rise to completely different cost functions for each style of design.

In this overview, we have explained the biases that are present in our overall project, and tried to give some feeling for the design process as we view it. We are attempting to utilize sophisticated software techniques to rapidly and correctly specify the various design representations, and to complement these

techniques with high performance design verification tools. While our emphasis is on techniques for creating custom integrated circuit designs, we also believe that our investigations have provided a great deal of insight as to the relative use of these techniques as opposed to semicustom techniques, including standard cell approaches and gate arrays. The research results provided by this contract can thus serve as a fundamental basis for highly efficient custom integrated circuit design fully capable of producing high performance designs for complex systems. In the succeeding sections of this report we describe the projects undertaken during this contract and report on our progress in the context of this overview. Thus, we hope to clarify the intention of our various activities and show how they lead to important design system methodologies.

Statement of Work

In this section we use the statement of work contained in our proposal to describe the basic goals of our research during this contract period. There are a large number of distinct goals, and we will first describe them separately, and then explain their interrelationship.

For some time we have been studying the use of constraints in a number of different ways within a design system for custom integrated circuits. There have been two major uses of constraints that have been explored. First of all we have devoted considerable effort to utilizing constraints as a basic declarative form of specifying the functionality required of a design. That is, if we state, in the form of constraints, all those properties that must hold in a particular system, then we have specified the functionality that that system exhibits. In this way constraints can be used to specify algorithmic confidence, and a substantial effort by Prof. Sussman and his graduate student Guy Steele was made in this direction. As a result of their work it was possible to demonstrate that classes of simultaneous linear equations could, in fact, be represented as a set of simultaneous constraints, and that it was possible to erect a number of independent solution strategies on top of this basic declarative network. In this way the constraints represented all of the conditions that held among the variables of the simultaneous linear equations, but did not specify the way in which these equations should be solved. While it was possible in this way to adequately

represent the competence of these equations, it has not been possible to extend these ideas further into other significant classes of problems. Accordingly our attention to constraints has turned more to their use within a design system to automatically derive values for variables that are stipulated by constraints, and to maintain the constraints through so called "truth maintenance" techniques. A simple example of this idea can be seen in the sizing of a transistor. An important parameter of each transistor, particularly for NMOS designs, is the length-to-width ratio. If we consider separately the length, width, and length-to-width ratio of an individual transistor channel, then it is clear that only two of these quantities can be set independently, and the third is constrained in terms of the other two. There are two benefits from the use of truth maintenance schemes in connection with such interrelated variables. First of all, it is clearly possible to propagate values for these variables when sufficient other parameters have been specified to force their values. Thus if the length-to-width ratio is specified, and the length is also given, then the system can automatically provide the width variable value. Not only is it convenient to have these values automatically supplied, it is also of course beneficial to have it done correctly in terms of the specified constraint. Another important virtue of constraints, however, is that they may be used to check on the mutual consistency of the values of sets of variables. Thus if a designer independently specifies values for the length, width, and length-to-width ratio of a particular transistor, and these values are not mutually consistent, then a constraint orientated truth maintenance system can ascertain this fact and advise the designer of this inconsistency. This is a great help of course and these ideas are easily extended to basic notions involved in design rule checking. Truth maintenance for constraint systems can obviously be used interactively in a design system to check that new design information that is added to the system is consistent with all of the spacing constraints required by the underlying process technology. These techniques have proved particularly useful in two of our programs. One of them, called Daedalus, is an interactive graphic system based on the MIT Artificial Intelligence Laboratory LISP machine which uses the constraint propagation ideas locally. It has become apparent over time that constraint propagation is not useful over a large design but instead is most useful for local adjustments within a small area of the overall design space. Another

program using the constraint propagation ideas is the text-orientated layout language DPL. This layout language provides the designer with a means for specifying constraints which he wants maintained and the system will automatically keep the variable values aligned in a manner consistent with this specification. Both of these systems have been very successful, and have received wide use within our design community.

Building on the strength of these ideas for local constraint maintenance, we have also started an investigation of the use of these ideas in logic and timing simulators. Clearly a network of logic gates constrains the values of the inputs and outputs of this gating structure. In fact the truth table for any given logic function summarizes the nature of these constraints. Our interest has been to use the constraint ideas to propagate logic values through a network to both control and observe values at a particular point of the network. Clearly this simulation technique is fundamental for testing, and we have been interested to see how we can apply the basic programmatic techniques for constraint propagation in this important area. Similarly we have examined the use of constraints for timing simulators. Constraints for the delays through any particular path of a circuit can be specified by the designer and tested for within a timing simulator. While we are just starting our investigation of this important area, we feel that these techniques should be basic to the avoidance of race conditions within a well designed circuit. In this way we hope to bring together our extensive work in performance analysis specification with this basic set of programmatic capability orientated around constraint maintenance.

In order to efficiently design complex systems, we are investigating compilation techniques that convert a high level linguistic specification for a design into a target machine which is constrained in terms of particular architectures. Two different kinds of architectures have been studied. On the one hand, we have extensive experience with microprocessor type designs as evidenced in the case of the SCHEME chip, which has just recently completed its second design iteration. This chip is motivated by the desire to provide a custom architecture to efficiently execute a dialect of the programming language LISP called SCHEME. Extensive knowledge of the way in which the SCHEME language statements are interpreted has permitted a highly optimized design both as far as the control is concerned and as far as data path

organization is concerned. A number of very interesting procedurally based compilation techniques have been developed for this purpose. And this continues to be a major thrust of our effort particularly within the artificial intelligence area. An increasing focus of our effort has been on the design of custom integrated circuits for signal processing, particularly since there is a large possibility for exploiting parallelism, and because of the need for extensive arithmetic capability in such designs. The MACPITTS compilation scheme is an example of such a compilation procedure, which utilizes a number of distinct finite state machines realized in terms of program logic arrays coordinated to a single shared data path. We have also studied a number of custom architectures for signal processing, and have completed the design of a high performance signal processing chip initially motivated by the needs of speech synthesis, but useful for a wide variety of filtering applications. It is our belief that experience with specific designs of this sort will help to steer and motivate our research for compilation techniques that will yield high performance signal processing designs. For example, an ongoing project derived from the design of the speech synthesis signal processing chip has been the provision of the programmatic capability for generating the layout of high performance parallel multipliers given the number of bits of the multiplier and multiplicand. In this way the designer can quickly generate large high performance multipliers without any need to specify detailed circuit and layout considerations. We believe that this kind of capability is unique and has not been provided previously and can be exceedingly useful in a building block approach to the assemblage of complicated digital signal processing systems. Given further experience with these individual blocks, they can then be called upon by a higher level compilation strategy and it is this direction that we are following. We are also exploring techniques for systematically examining the different degrees of parallelism that are available in a digital signal processing system. Given one particular design, we already have available formal techniques for manipulating that design into other versions of the design that make a different tradeoff between space and time. The coupling of this high level means for exploring architectural choices with the algorithmic basis for generating the individual cells will give rise to a comprehensive system for the design of digital signal processing applications.

An important aspect of our work has focused on the provision of high

performance design rule checking algorithms. There are several aspects to this work. Firstly, we have examined two different kinds of algorithms. The traditional design checking algorithm represents the layout information in terms of a set of rectangles. On the other hand, mask information can be specified in terms of a symbolic coarse grid, and this gives rise to highly regular algorithms for design rule checking. We have constructed design rule checkers using both of these data representations and found that they each have their place utilizing different kinds of technologies. If the design rule checking algorithm is to run on a standard conventional computer then the rectangle based algorithm has proved to be most satisfactory. The particular version that we have constructed is modular in the sense that it is easily adapted to a variety of different technologies, and by careful attention to sorting and sequencing, we have been able to speed up the performance between one and two orders of magnitude. Also, for sparse designs, since the design rule checking time is roughly proportional to the square of the number of rectangles, this rectangle based algorithm can be quite fast. In contrast to this technique we have also designed a design rule checking algorithm that first rasterizes all of the mask information, and then checks the design rules by using pattern matching techniques over the coarse grid. The time needed for this technique is independent of the number of rectangles in the design, and uses an exceedingly regular computational structure. For this reason we have been studying for some time the design of special purpose hardware for performing design rule checking using these techniques. This has necessitated the design of four custom integrated circuits, but each of these is highly regular itself and is not large. These four chips, together with several other support chips, will be mounted on a single microprocessor-based board which can serve as a plug-in functional unit for an individual work station. In this way very high speed design rule checking can be provided for substantial designs on an interactive basis. The study of both of these techniques has benefited from our concerns with the use of hierarchy and design rule checking. A number of different hierarchical strategies have been proposed, and the special purpose hardware treatment just described does in fact utilize a hierarchical approach. The basic idea behind this hierarchical approach is to treat every cell of the design as being completely well formed and hence completely checkable by itself. When these individual cells are combined to form larger units, then it is only necessary to check the

satisfaction of design rules along the interstices of the design where the cells come together. This approach has been very successful and is now being applied to all of our design rule approaches. Since we have completed our design for the special purpose hardware for design rule checking, and are currently implementing this design, we have also turned our attention to the use of such techniques for artwork analysis for other tasks such as topology extraction. As these algorithms are developed, we are confident that more general architectures for artwork analysis will evolve that will perform a number of these tasks efficiently and flexibly.

A major facet of our work has been concerned for circuit performance issues in terms of speed and power. A major tool in these studies is of course, circuit simulation, which has been available for some time, but which is computationally expensive. We have thus sought to bridge the gap between the detailed mask layout information and the input format needed for the popular SPICE circuit simulation program. Accordingly, we have designed a circuit extraction algorithm which takes as input the detailed mask specification and provides as output that information needed by the SPICE input deck. Thus the user of this simulation program is spared the need to understand this input format and can automatically derive all of the needed parameters from the layout by simply invoking one program. In the past, circuit extraction programs have been available, but they have not been very accurate. We believe very strongly that as the scaling of layout dimensions goes on leading to smaller and smaller widths and spacings, that highly accurate resistance values and internodal capacitances will be needed in order to afford an accurate circuit simulation. In the program under development, it is possible to calculate resistances either by counting squares, by using form factors, or by solving Laplace's equation for the resistance of nonstandard shapes. Thus within the confines of one well structured program the user is able to obtain those computational resources needed to calculate resistance values associated with arbitrary forms in the layout. In the case of capacitances, not only are capacitances to the substrate computed, but internodal capacitances, even when the nodes are substantially distributed in space, are calculated. These values are particularly important for dealing with noise problems due to coupling between conductors in a dense integrated circuit. This program has proved to be exceedingly valuable, and is undergoing continuing revision using recursive techniques for automatically gaining a prespecified degree of

accuracy for both resistances and capacitances. It also turns out that the program may be used in a simpler mode to obtain the network topology of the circuit specified by the mask layout. This information is necessary for logic simulation, and in fact a new unit delay logic simulator has also been written that couples with this program in a very efficient way. A new area of study that is complementary to these concerns is the examination of circuit simulation techniques that are both accurate and fast, there has certainly been substantial progress in this area within recent years, but we believe that by the study of new modeling techniques coupled with innovative control strategies that it will be possible to provide, possibly with the use of special hardware, very high performance circuit simulation that is also very accurate. Certainly without such resources, it is impossible to design high performance circuits.

Once accurate circuit modeling values are available, it is of course important to investigate the performance of these circuits. We have been particularly interested in studying the delay through interconnect as evidenced by long polysilicon wires that are frequently the source of substantial limitations on performance in large circuits. It turns out that analytical relations can be used to bound delay through such interconnect, and that these relationships can be computed in a very efficient way. Thus we hope to be able to incorporate these techniques within a design system that will allow a rapid interactive exploration of timing and clocking structures. We are also working in this connection on techniques for optimally sizing transistors in order to fulfill speed and power requirements. Given a chain of such devices, it is possible to compute the optimal size for each transistor according to some stipulated criteria. We believe that it is also possible to use dynamic programming techniques to both discover and characterize maximum delay paths within any given network.

For some time we have been working on the design of a large program to automatically place and route a set of rectangles of arbitrary aspect ratio but orthogonally related with interconnect on all four sides. The provision of a program to solve this problem is a large undertaking, and has involved a great deal of study in both global and local routing techniques, including channel routers and power distribution techniques. We have been working on this problem for several years, and we are now at a stage where the program is

performing adequate routing capability, but the placement techniques are not yet completely incorporated within the overall program. The design of this large program has brought into focus a number of design issues and given rise to many different student projects. Our approach is algorithmic, and does not contemplate interactive use by the designer. Nevertheless, as we apply this program to a number of different designs, it may be that some sort of interactive capability is desirable. We emphasize that this effort is focused around the derivation of optimal algorithms, and that the achievement of such a program will be a major accomplishment since it is widely recognized as a basic requirement for all comprehensive design systems.

In the previous paragraphs we have discussed many different programs that deal with a variety of different representations as well as the transformations between these representations. As we have noted before, there are two basic computational environments that we use for the development of this software. By far the largest number of programs are written for a DEC system 20 which gives us a great deal of flexibility in programming languages, and also provides for time shared use by great many researchers. This is also the facility that is used to support our classes in integrated circuit design, so there is a natural means for providing us with user experience in association with all of our programs. The second computational environment that is heavily used is that of an MIT Artificial Intelligence Laboratory LISP machine, coupled with a high performance color display as well as a monochromatic high resolution display. This system has been used for the development of the Dedalist system, and most of the LISP based tools such as DPL. Both of these computational environments are connected together on a local network which is also connected to a great many other research facilities so that access is very flexible. An increasing emphasis of our work is to consider the design of a unified data base that will be appropriate for all of the different programs involved in the design process. This is a large undertaking, and needs to be approached cautiously, particularly since many of the programs are in a continuing state of flux and it may be undesirable to constrain them to dealing with a rigidly formulated data structure. We see this problem as an inevitable tension arising from undertaking research on the design of custom integrated circuits. On the one hand, we need the flexibility to innovate new solutions to a wide variety of problems, on the other hand, we need to continually examine how these

resources can be interconnected to provide a unified user orientated system. Nevertheless, we feel that it is time to explore the problems of a coordinated system, and this will certainly be an increasing part of our effort.

We have mentioned in the paragraphs above that there are several different designs that have motivated our creation of design tools. Most important of these are our work on the SCHEME chip and our digital signal processing chip. As we have commented, these designs are of widely different nature, and yet we have been able to learn a great deal from these efforts. There are also, of course, a wide variety of other designs going on, and we are currently exploring the possibility of the design of a chip to be placed in a large network appropriate to a "connection machine" suited for a large artificial intelligence problems. This project will raise additional issues having to do with a heightened level of communication between neighboring chips and system coordination issues involving literally hundreds of processors. Thus, we continue to expand the variety of designs that we have experience with, and this enriches our confidence as to the worthiness of the design tools being developed.

In the paragraphs above we have summarized the work undertaken for this contract, and placed it in the context of our overview. We have tried to give a feeling for the evolving direction of the work, as well as the current status. In the next section we deal explicitly with results obtained during this contract year.

STATUS OF RESEARCH

In this section we give a detailed view of the ongoing status of our research together with concrete accomplishments. A substantial number of publications have resulted from this work and are listed separately at the end of this report. Our work has been appropriately represented at the conferences germane to this field, most particularly the Design Automation Conference held each year in June. During this last contract year three papers were given at the Design Automation Conference representing work sponsored by AFOSR and one of them ("A "Greedy" Channel Router" by R. Rivest and C. Fiduccia) won a best paper prize. Furthermore all of our programs are widely distributed and are available within the United States at no cost. We believe that this policy has resulted in widespread use of our programs and a lot of valuable feedback.

For example, virtually every university carrying on substantial research in integrated circuit design is using the MOSSIM unit delay logic simulator developed by Randal Bryant under this contract.

The point of view taken by our research has been expressed in a recent review article, written by Professors Allen and Penfield titled "VLSI Design Automation Activities at MIT". This article provides both a basic conceptual point of view and also a description of the various research projects going on at MIT in the design automation area, almost all of which are supported under this contract.

In previous reports, we have put a great deal of emphasis on our research on constraint representations and the means of exploring various performance alternatives at the architectural level. During this contract year, less basic research is going on in this area, but the ideas previously generated are being used in a number of different ways. For example, the idea of exploring various space/time tradeoffs at the architectural level has been utilized in the MACPITTS project by designing a high level functional specification language in a way that permits the user to explicitly control these tradeoffs. What is missing, however, is the means to make these conversions automatically in a semantically in variant way. That is, although we know from previous research precisely how to control changes so that the underlying competence remains invariant, we have not yet implemented this in a programming language. This is part of our ongoing research, and the issues are well understood, so the remaining task is to implement these ideas within our design software. The constraint ideas which we have previously discussed, are also being heavily used, particularly in the Daedalus system and the DPL layout language, both of which encourage the user to use truth maintenance and constraint propagation techniques for maintaining the well-formedness of the growing design.

A great deal of attention has been devoted to the creation of effective mask layout techniques. We have already mentioned the DPL text-based language, that provides for symbolic manipulation of important design variables, as well as the daedalus system which is capable of converting a graphic layout into a DPL orientated data base. Another layout language which has been used here for some time is the AIDS language which is implemented in APL. AIDS is a

well constructed program, and was the first interactive layout language available in our community, but since it runs interpretably, it is computationally expensive for large designs. Nevertheless, largely through Professor Penfield, there is substantial APL expertise here and this language represents an important approach to the layout problem. By far the most popular interactive graphic layout language presently used here is the program HPEDIT which utilizes the HP 2647 or 2648 graphic terminals. This program is self documented, menu driven, hierarchical and capable of zooming in on an appropriate level of detail. Users find that they can learn to use this language very quickly, and since it is written in C, it compiles very efficiently on a number of different machines and has been used at many different locations. It has been readily adapted from NMOS into CMOS, and a recent improvement has been to provide interactive design rule checking within the program. HPEDIT has been rewritten at least twice, so it is now a much more reliable program, and much easier to modify. It is hard to over-emphasize the importance of the human factors considerations that have been used in the design of HPEDIT. This is particularly important in our environment where we have many users who must learn to use the software quickly, and renew their understanding after a lapse of time with minimal cost in time. Given an improvement in our facilities base, we will probably extend HPEDIT to color graphic terminals, but this should present no major new difficulties. We have also developed a small program called MSHOW which allows the user of a conventional alphanumeric terminal to display coarse layout information. The display generated on such terminals is not very desirable, but when the alphanumeric terminals are the only ones available, it has been found to be very useful. No great effort has been put into this program, but it will be maintained in order to provide a certain basic low level capability from any conventional terminal.

We are particularly pleased with our activity in placement and routing. Under the direction of Professor Rivest, a lively research group has sprung up to consider all of the algorithmic problems involved in this task. One of the most interesting projects was the development of the so called "Greedy Channel Router" which provides optimal channel routing capability for custom integrated circuits. A number of extant channel routers were studied and the techniques used by them were examined in the context of many different designs. From these studies, a set of elemental strategies evolved that have

proven capable of spanning all of the different strategies required in channel routing. A control structure was then invented to coordinate these basic moves, and a great deal of testing was done in order to optimize this high level strategy. The result has been a high performing channel router, but one which is also transparent to the user. That is, it is possible to understand the basic moves used by this program, and of course it is an easy matter to modify the control structure based on increased understanding through experience. It is probably the insightfulness that has been brought to bear in this router that contributed in large measure to the best paper prize that was awarded to this work at the 1982 Design Automation Conference.

Channel routers, of course, provide only one ingredient in an overall placement and routing strategy. The major task faced in this area is the creation of a complete system for placing and routing rectangles of arbitrary aspect ratio orthogonally related in minimal area. This has become a classic problem, and is being studied by a large number of groups, but we feel that we have achieved a leadership position in this area. By carefully combining both global and local strategies, and using appropriate theoretical techniques for the generation of spanning trees and other needed routing structures, it has been possible to develop a well motivated strategy. The system that we are developing, called PI (for placement and interconnect) is still under development, but it is already performing at a good level for interconnect. Substantial problems remain for dealing with placement issues, and major emphasis is being put on these issues. It may be that routing should be treated as requiring an expert system, and that even several kinds of experts are needed for different kinds of routing, such as "river routing". If such capability were available, then the designer might ask the expert program to perform certain routing tasks and the program might maintain a dialogue with the user as to the appropriateness of such a move, the degree of difficulty, the completeness of the routing, and the cost in space. Such an interactive approach is in contrast with the more algorithmic approach used by the PI system, and both of these techniques are currently under study. It is a hallmark of our approach to many problems that we are willing to entertain several different solutions to the same problem. Thus, we always have a substantial level of internal competition between alternate approaches, and this is very useful. In the routing area, the assessment of goodness of the various approaches can be ascertained by testing on standard bench mark

problems, and these have recently been collected in a useful documented form for all of those working in this area.

Continuing at the artwork level, we have for several years been concerned with the efficient implementation of design rule checking programs that are suitable for a variety of technologies. We have from the start restricted these strategies to those layouts that incorporate mainly orthogonal geometries with a possible inclusion of 45 degree lines. No other arbitrary shapes have been allowed, since the cost of including these in any practical design rule checking program is exorbitant. Our initial experience has been with rectangle-based design rule checking programs and several of these have been built. Perhaps the most flexible has been a program, written in C, which provides a number of basic operations, as might be conceived of as available in a virtual machine designed for design rule checking, which can then be combined by the designer to formulate arbitrarily complex design rules. Thus a basic set of capabilities can be drawn upon to easily construct design rule checks, without the need for the designer to deal with all the combinatoric complexity involved in managing the thousands of rectangles present in any realistic design. This design rule checker, initially implemented at MIT Lincoln Laboratory, has been substantially improved through careful study of the scanning and sorting routines that often dominate the execution time of such programs. Almost two orders of magnitude improvement in time have been generated through this careful reexamination, and this has been enough to make this program exceedingly useful on an interactive basis within the HPEDIT interactive layout language. Thus, a medium size circuit can be readily checked in a matter of a few seconds, and this capability has proved to be exceedingly popular with our user community. A different tack on the design rule checking problem is provided by first rasterizing the design into a coarse grid of symbols, each of which represents the masks present in one of the grid squares. Design rule checking can then be done by using pattern matching techniques associated with a moving window four squares on a side. Such an approach leads to an exceedingly regular program design, the execution time of which is proportional to the area of the chip rather than the number of rectangles present in the layout specification. This regularity has suggested the use of special hardware in order to improve the execution speed, since no conditionals are present in the code and hence pipelining of activity can be used without penalty. Accordingly, a major project of ours has been to

design such a special purpose piece of hardware, which is a board level system containing four custom integrated circuits. All of these four circuits have been designed, and two of them have been returned from fabrication without any error at all. When this board level product, including microprocessor control, is completed during the next contract year, we expect to achieve at least two orders of magnitude speed up in the overall processing of design rules. For large chips, this capability is indispensable, and should be thought of as providing a special functional unit for an interactive VLSI work station. We have given papers on this work at two conferences, and the work has been very well received.

One of our earliest programs was MOSSIM, which is a unit delay logic simulator. The trajectory of this work through time has been very interesting, since the original work was focused mainly on providing a switch level simulation, but as a result of continual refinement, a basic theory for MOS digital systems has evolved. The earliest work revealed that gate level simulators were inappropriate for MOS design, and hence our logic simulation investigations focused at the individual transistor or switch level. This decision has proved to be very helpful, and no design generated here is ever dispatched for fabrication without a complete logical checking at this level. The switch level seems to be an appropriate level of abstraction for MOS designs and captures important circuit activity leading to the desired logical behavior. What is most remarkable about this work is that the implementation of the simulator itself led to a thoroughgoing examination of the basic logical foundations for MOS digital circuits, and a theoretical analysis and model has been provided for this purpose in the doctoral thesis of Randal Bryant. Thus not only was a practical problem successfully attacked by the design of the simulator, but basic theoretical understanding was also provided in a complementary fashion. We feel that this style of attack on problems is particularly appropriate for university research, since an immediate problem is attacked and solved but the more fundamental issues are also appropriately addressed.

A great deal of our emphasis has been on circuit performance, and this emphasis can be expected to increase with time. The point of view taken by much of our research is expressed in an article on VLSI circuit theory by Professors Glasser and Penfield where the implications of the large complexity

of a design on the kind of circuit theory that must be developed is carefully treated. We place much emphasis on circuit description, because we believe that this level of representation is the heart of high performance in circuits which must not be neglected, particularly in the university setting. Many different topics have been investigated including the scaling of clock noise, the syntactic interconnection of circuits for well-formedness, and the development of circuits that can probe and test the performance of any given process. Two of our most interesting projects however, concern the extraction of circuit models from layout and the bounding of time performance along RC interconnection trees in MOS circuits, particularly those involving polysilicon lines. The circuit extraction work, which is instantiated in a CLU program, provides a transformation between the layout description and the input file for the SPICE circuit simulator. Highly accurate values of resistance and capacitance are computed with all parasitic effects being appropriately considered. Here again we place much emphasis on the need for accurate values in order to achieve tight bounds on the estimation of the performance of circuits. Users find it very useful to be able to go from a layout description directly into SPICE and to be able to access with great confidence the performance of their system prior to fabrication. It is hard to over-emphasize the utility of this capability, and its importance for modern design. This program is highly modular, and can easily be adapted for a variety of technologies. In another substantial research activity, involving three of our faculty, very useful models that bound the delay through RC interconnection paths have been produced. This is another example of bringing together highly fundamental and advanced theoretical techniques with very practical circuit considerations. This work has attracted a great deal of attention both at other universities and within industry and is a good example of new techniques that are available for dealing with the complexity of modern designs. We plan to continue to emphasize this work, and to incorporate it in timing verifiers but can give estimates of overall performance prior to fabrication as well as revealing possible race conditions. Although it is hard to point to a major research focus in this area, our interest in performance has led to the constant examination of new circuit forms in a variety of different designs. Without the kind of programmatic tools that we have just described, there is no question that our research community would not focus on these issues to anywhere near the extent

that has actually happened in practice. There is no question that in the future we will incorporate these techniques within rapid compilation strategies so that the performance consequences of a number of different architectural tradeoffs can be examined quickly and accurately. This is really the important direction that research in computer-aided design for custom integrated circuits must take. When the tools are not available to quickly and accurately explore tradeoffs, these tradeoffs are simply not examined. In a sense, then, we are moving in a direction where designs are debugged while they are still at the model stage rather than being constructed. This is entirely appropriate, of course, for custom integrated circuits and proper use of these techniques leads to high performance circuits at the time of initial fabrication. In the circuit area, we should also mention that we are shifting our emphasis from NMOS to CMOS. This has meant the development of a large number of new circuit forms, the adaptation of our existing tools to CMOS, and the provision of many library cells, such as pads, in this new technology. This work has actually consumed a considerable amount of time, and although it perhaps does not qualify as basic research, it is important to mention it here, since it has happened fairly easily and attests to the flexible nature of our design tools which have been readily adapted for this new use.

The area of applications has been exceedingly important to us, because it provides a test bed for a number of our design tools and architectural frameworks. Increasingly, a great deal of our emphasis has been placed on signal processing, since we need both architectural performance (i.e. parallelism) as well as circuit performance, in terms of minimal area, high speed, and minimum power. We have designed a high performance signal processing chip for speech synthesis, which includes a sixteen by twenty-four multiplier and a large amount of on-chip memory. This was a particularly interesting study because it derived from an earlier TTL design using serial arithmetic. Careful analysis, however, indicated that serial techniques were inappropriate when used with the MOS technology, and the resulting integrated circuit version has used a parallel architecture. The issues involved in assessing the conversion between two different technologies, particularly the conversion from TTL to MOS, has aroused a great deal of interest in industry, and we feel that a great many useful lessons have been learned from this experience. They have been documented in an article by Evans and Allen which

is listed among our publications. The intimate interrelationship of signal processing tasks with VLSI has been addressed by a number of our researchers, and this continues to give rise to the study of new architectures for highly parallel approaches to canonical signal processing problems. We are actively studying the design of parameterizable floating point units, which are just now coming into use, and are finishing the construction of a multiplier assembler which is capable of generating a high performance parallel multiplier where the word length of the multiplier and the multiplicand can be specified as variable parameters by the designer. The provision of complex number arithmetic and its consequences within linguistic frameworks has also been explored for APL with implications for VLSI design. The view that we have presented here of the interaction between signal processing and VLSI shows our intent to combine fundamental theoretical studies with practical implementation issues. Our research environment here is well suited for this approach, and we believe that it has given a great deal of strength to our work. Work also continues on the design of the SCHEME chip, which is now in its second version, and which provides a great deal of architectural parallelism for the efficient execution of the SCHEME dialect of LISP. A multiprocessor architecture has evolved which will be capable of using this chip together with many other specialized processors to produce high performance multiprocessors systems. We are also studying the design of a nodal processor for the very large connection machine being constructed in the Artificial Intelligence Laboratory. This project is just now under way, but raises interesting architectural issues as well as fundamental problems in testing.

The facility base for all of our programs has been evolving over time, although the diversity of programming languages being used is diminishing. Far and away the most heavily used facility is the DECSYSTEM 20 which supports at least half a dozen languages and almost all of our design software. This is a large work house facility providing time shared capability for a large variety of research, and is used in all of the different research laboratories here at MIT doing design research. Many of the programs are being written in C and in LISP and recently two small VAX computers have been added. Eventually these VAX systems may become part of the individual design stations suitable for highly interactive use. Of course the MIT Artificial Intelligence Laboratory LISP machines are also used for this purpose and they

provide a very rich software environment in LISP plus highly interactive graphic capability all of which is integrated together into a very powerful system. We have some capability on the LISP machine and also on the VAX systems for color graphics, but this capability will improve with time. As these facilities evolve toward individual graphic design stations, there is no question that increased emphasis on our work will be focused on coordinated data bases and well defined interactions between all the pieces of our software. Our approach has always been to develop the individual programs first and to then examine the way in which they can be coordinated together into a larger system. This is a bottom up approach, and we believe the time is now approaching where a top down coordination strategy should be developed and may be beneficial to the overall performance and user convenience of these design stations.

IV. PUBLICATIONS

In the list below, we cite the publications and internal memos that have been written under the sponsorship of this contract, and which describe the work done during this contract period. Readers wishing additional detailed information on any activities conducted under this contract should consult directly with Prof. Allen.

1. Evans, W.H. and Allen, Jonathan, "MOS Implementations of TTL Architectures: A Case Study" Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, Paris France, May 3-5, 1982.

2. Penfield, Paul Jr., "Small is Big: The Microelectronic Challenge" Proceedings, The Innovative Process: Evolution vs. Revolution, MIT Industrial Liaison Program Symposium for Senior Executives, London, England, November 12-13, 1981.

3. Penfield, Paul Jr., "AIDS, APL Integrated-Circuit Design System" Proceedings APL 81, October 21-23, 1981, San Francisco, Cal.

4. Penfield, Paul Jr., "Principal Values and Branch Cuts in Complex APL" Proceedings APL 81, October 21-23, 1981, San Francisco, Cal.

5. Glasser, L. A., and Penfield, Paul Jr., "VLSI Circuit Theory" Proceedings Large Scale Systems Symposium, October 11-13, 1982, Virginia Beach, Va.

6. Penfield, Paul Jr., and Rubinstein, Jorge, "Signal Delay in MOS Interconnections" Proceedings of the Second Caltech Conference on VLSI, January 19-21, 1981, Pasadena, California.

7. Kopec, Gary E., "The Impact of VLSI on Signal Processing Algorithms and Architectures" Trends and Perspectives in Signal Processing Vol. 1, No. 3, July 1981.

8. Bryant, Randal E., "MOSSIM: A Switch-Level Simulator for MOS LSI" Proceedings, 18th Design Automation Conference, Nashville, Tenn., June 29-July 1, 1981.

9. Bryant, Randal E., "A Switch-Level Model of MOS Logic Circuits" VLSI 81, John Gray ed., Academic Press 1981, pp 329-340.

10. Seiler, Larry, " A Hardware Assisted Design Rule Check Architecture" Proceedings, 19th Design Automation Conference, Las Vegas, Nevada, June 14-16, 1982.

11. Seiler, Larry, "Special Purpose Hardware for Design Rule Checking" Proceedings Second Caltech Conference on Very Large Scale Integration, Pasadena, California, January 19-21, 1981.

12. Rivest, Ronald L., "The "PI" (Placement and Interconnect) System" Proceedings, 19th Design Automation Conference, Las Vegas, Nevada, June 14-16, 1982.

13. Rivest, Ronald L., and Fiduccia, Charles M., "A "Greedy" Channel Router" Proceedings, 19th Design Automation Conference, Las Vegas, Nevada, June 14-16,1982.

Internal Memos:

1. Penfield, Paul Jr., "Signal Delay in RC Tree Networks" January 1981 VLSI Memo 81-40.

2. Kopec, Gary E., "MSHOW (multi-media layout dispay program)" July 1981 VLSI Memo 81-53.

3. Glasser, Lance A., "Clocking Semi-Groups for VLSI Circuit Analysis" September 1981, VLSI Memo 81-63.

4. Glasser, Lance A., "The Syntactic Analysis of VLSI Systems Using Graphs" September 1981, VLSI Memo 81-62.

5. Glasser, Lance A., "A Canary Straw Bird" February 1982 VLSI Memo 82-79.

6. Rivest, Ronald L., ""Benchmark" Channel-Routing Problems" February 1982 VLSI Memo 82-77.

Thesis:

1. Bryant, Randal E., "A Switch-Level Simulation Model for Integrated Logic Circuits" MIT Department of Electrical Engineering and Computer Science Ph.D., March 1981.

V. PROFESSIONAL PERSONNEL

Profs. J. Allen

L. Glasser

P. Penfield

R. Rivest

G. Sussman

Dr. H. Shrobe

EN
DA
FIL
5
DT